# Anomaly detection on Hypothyroidism dataset

Unsupervised Learning 2023-2024

Presentation

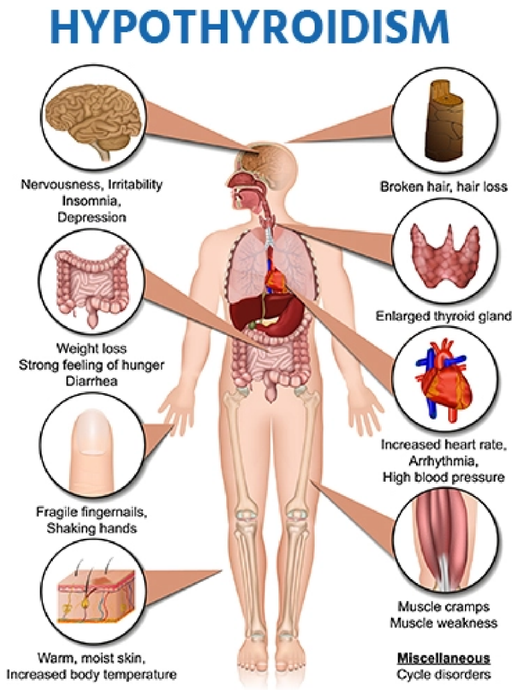
MSc Artificial Intelligence for Science and Technology

Unsupervised learning

Submitted by **Daniele Cecca Mat.918358**

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

# Introduction




This project focuses on the development and implementation of an **anomaly detection system for a hypothyroidism dataset.**

Hypothyroidism is a common condition where the thyroid doesn't create and release enough thyroid hormone into your bloodstream. This makes your metabolism slow down.

Also called underactive thyroid, hypothyroidism can make you feel tired, gain weight and be unable to tolerate cold temperatures.

The goal of this project is to identify **strange pattern in medical data that may flag potential cases of hypothyroidism** that require further investigation.

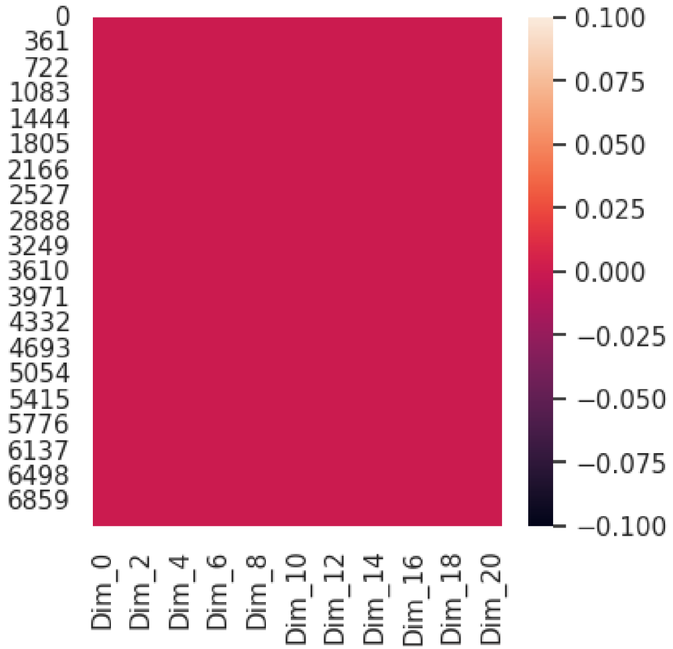
Submitted by Elvazta Rimberio



# PROJECT STRUCTURE

| | | |
|---|---|---|
| 1 | **DATA EXPLORATION** | In the first phase we did **analysis of the dataset**, including **data preprocessing**, handling **missing values**, **duplicates** and visualizing **data distributions.** |
| 2 | **ANOMALY DETECTION** | We apply four different anomaly detection algorithms:**DBSCAN LOF AUTO-ENCODER MCD** |
| 3 | **CLUSTERING** | The we did a comparison and an analysis of the anomalies detection models by using different metrics and **K-Means.** |
| 4 | **PROBABILITY** | At the end we will establish the **probability of each sample of being an outlier**. |

# Dataset Exploration




The dataset is compose of **7200 samples** and **21 attributes**:

- **15 binary**
- **6 continuous**

There are **0 missing values.**

There are **71 duplicates.**
We decided to remove all duplicates and keep only the first occurrence of each sample.

Submitted by Daniele Cecca

# Univariate analysis

## CONTINUOUS ATTRIBUTES

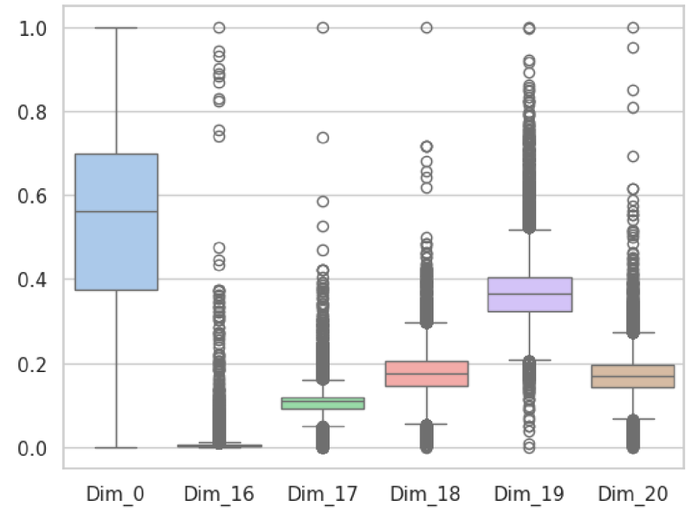
| | Dim_0 | Dim_16 | Dim_17 | Dim_18 | Dim_19 | Dim_20 |
|---|---|---|---|---|---|---|
| count | 7129.000 | 7129.000 | 7129.000 | 7129.000 | 7129.000 | 7129.000 |
| mean | 0.532653 | 0.009227 | 0.108475 | 0.179624 | 0.374209 | 0.173789 |
| std | 0.197311 | 0.043569 | 0.042208 | 0.060442 | 0.088791 | 0.056677 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.375000 | 0.001321 | 0.091922 | 0.145485 | 0.324074 | 0.143750 |
| 50% | 0.562500 | 0.003208 | 0.109192 | 0.173913 | 0.365741 | 0.170313 |
| 75% | 0.697917 | 0.005094 | 0.119777 | 0.205686 | 0.402778 | 0.195313 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

To analyze the continuous attributes we start by computing some **basic statistics.**

All the continuous attributes are within the same interval **[0,1]** and have similar variance and mean.

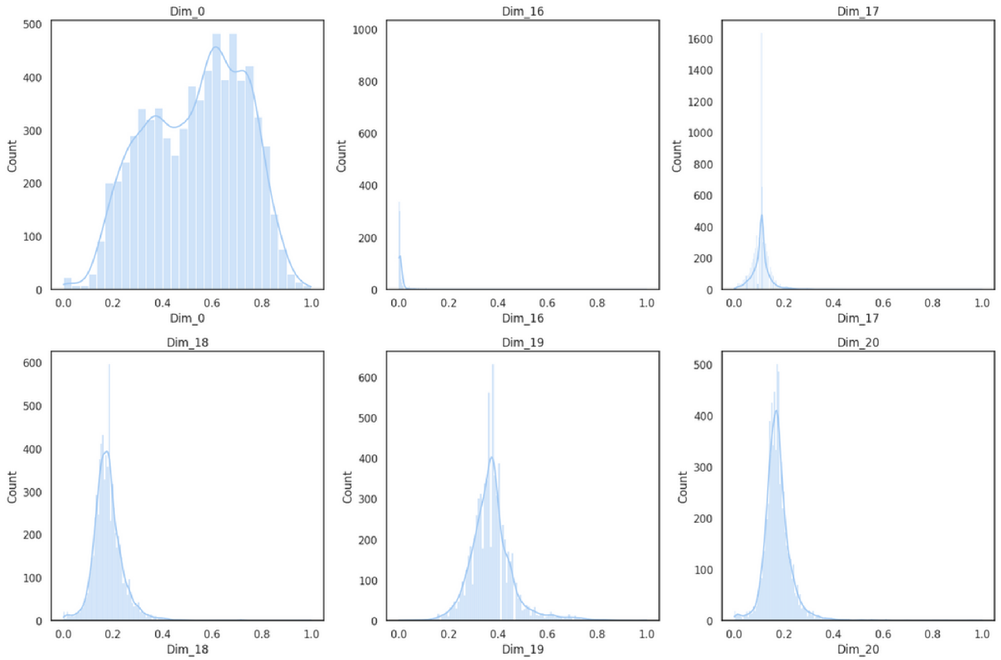
By analyzing the **box plots**, we can deduce that we have many **possible outliers**, at least from a statistical point of view.

we can already say that distributions likely have **heavy tails**, indicating that it probably does not follow a normal distribution.



| Feature | Skewness | Kurtosis |
|---|---|---|
| Dim_0 | -0.219 | -0.873 |
| Dim_16 | 14.454 | 259.349 |
| Dim_17 | 3.473 | 45.395 |
| Dim_18 | 1.510 | 11.000 |
| Dim_19 | 1.243 | 4.821 |
| Dim_20 | 2.373 | 22.437 |

From the results, we can conclude that they are **slightly asymmetric with heavy tails**, indicating that they do not follow a normal distribution




# Univariate analysis

## CONTINUOUS ATTRIBUTES

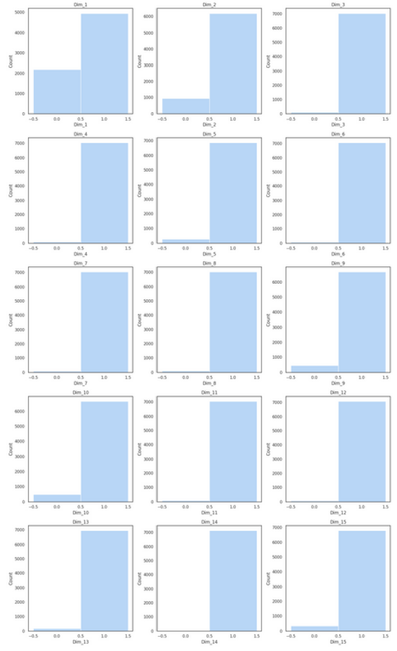
To validate our earlier hypothesis, we examine the distribution through histogram visualization and computation of **Skewness and Kurtosis**

# Univariate analysis

## BINARY ATTRIBUTES

To inspect binary features we count the number of occurrence for each variable and we plot the histogram.

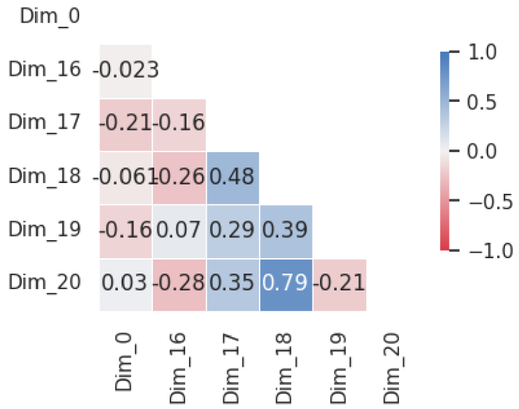
As shown in the results, we have a **smaller number of 0s compared to the number of 1s**

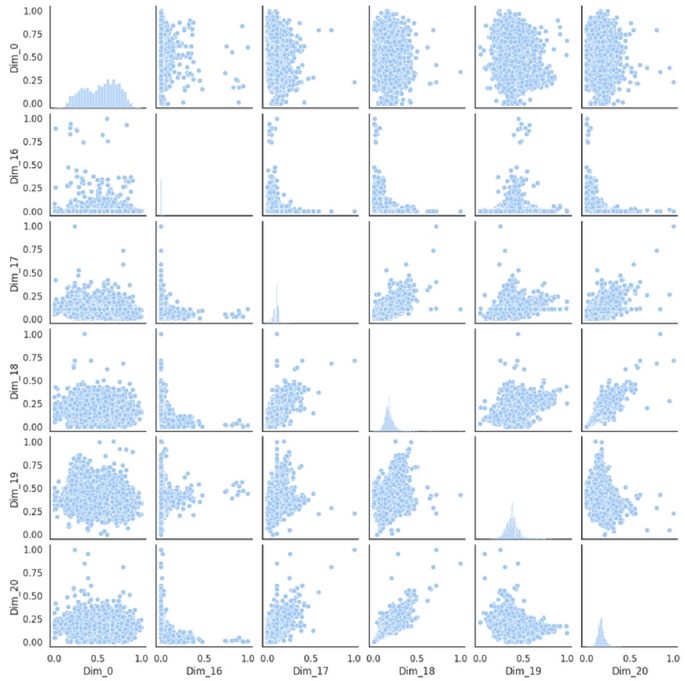


| | Value 0 | Value 1 |
|---|---|---|
| Dim_1 | 2187 | 4942 |
| Dim_2 | 940 | 6189 |
| Dim_3 | 111 | 7018 |
| Dim_4 | 91 | 7038 |
| Dim_5 | 276 | 6853 |
| Dim_6 | 78 | 7051 |
| Dim_7 | 101 | 7028 |
| Dim_8 | 121 | 7008 |
| Dim_9 | 472 | 6657 |
| Dim_10 | 492 | 6637 |
| Dim_11 | 91 | 7038 |
| Dim_12 | 59 | 7070 |
| Dim_13 | 184 | 6945 |
| Dim_14 | 1 | 7128 |
| Dim_15 | 352 | 6777 |

# Bivariate analysis

## CONTINUOUS ATTRIBUTES



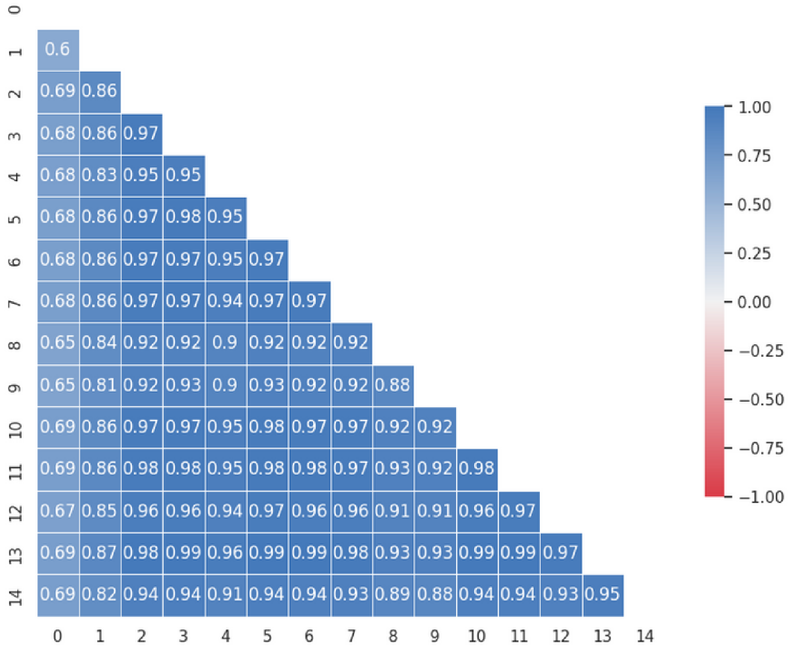




To capture the relationship between attributes, we could utilize **correlation analysis**

In this case, we decide not to standardize the variables because **correlation is scale and location invariant,** and also because, as we have seen before, they are within the **same interval.**

# Bivariate analysis

## BINARY ATTRIBUTES



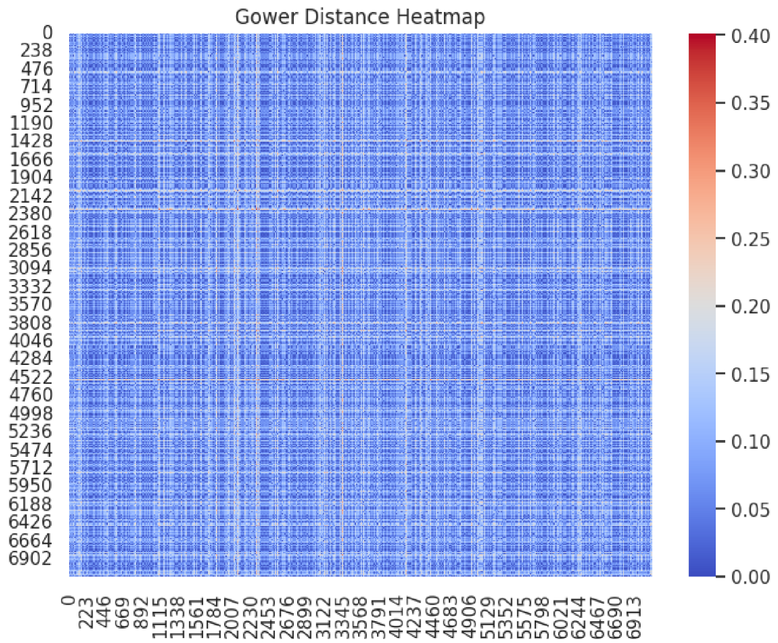

To compare two binary attributes, we can use the **Jaccard Coefficient**. If the value is equal to 1, it indicates that the two attributes provide the same information, and we could consider removing one of them

We don’t know what these variables correspond to, we prefer **not to remove them**

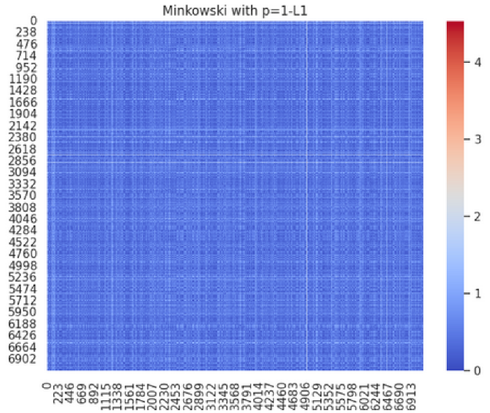
Submitted by Daniele Cecca



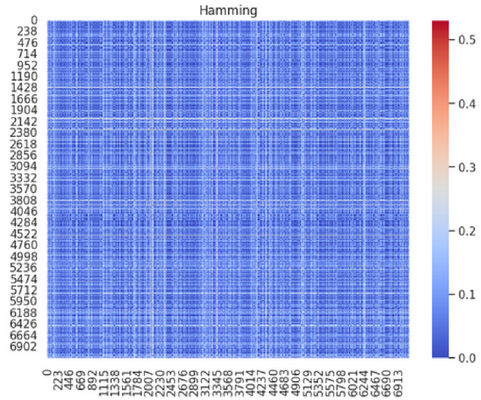
## GOWER DISTANCE




## MANHATTAN DISTANCE




## HAMMING DISTANCE



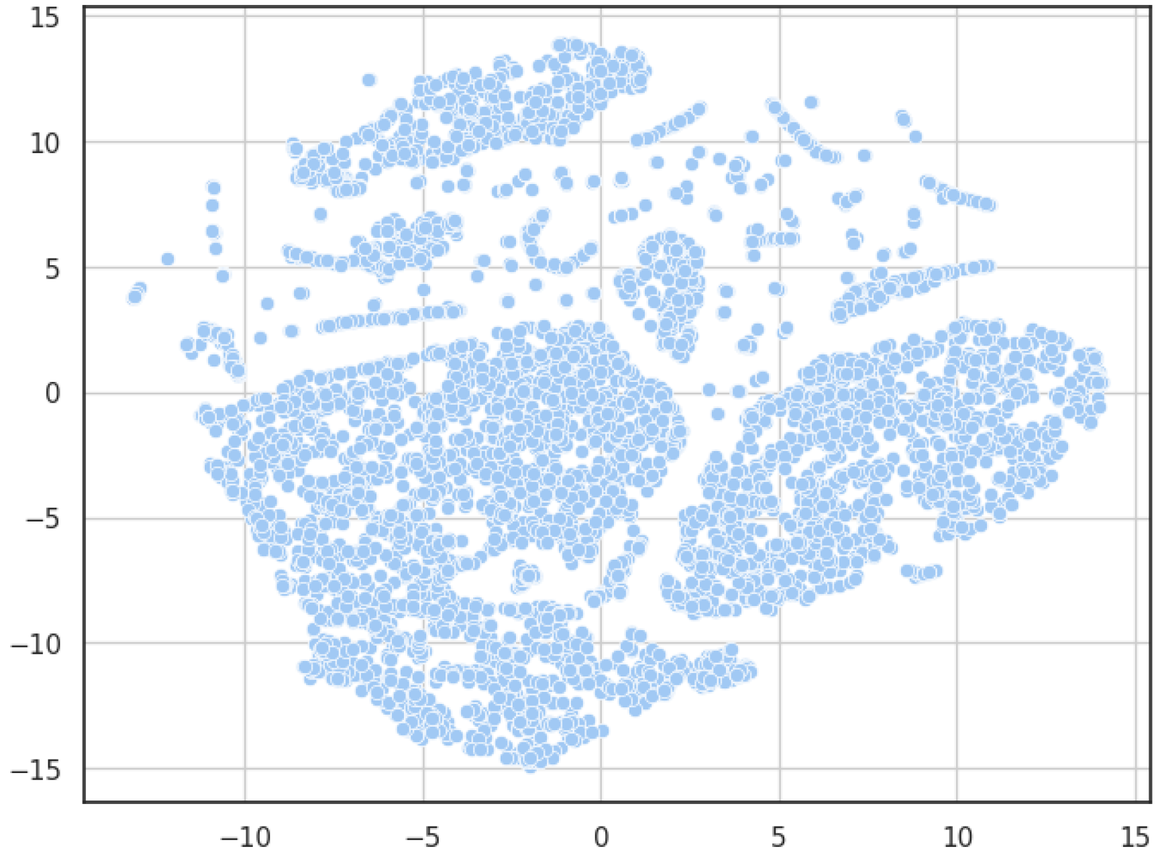

# Sample analysis

We **need both continuous and binary variables** to detect outliers because both of the results doesn't give enough information.

Submitted by Daniele Cecca

# Dataset visualization

We use t-Distributed Stochastic Neighbor Embedding **t-SNE**



# Anomaly detection

To detect the anomalies we will use four different approaches:

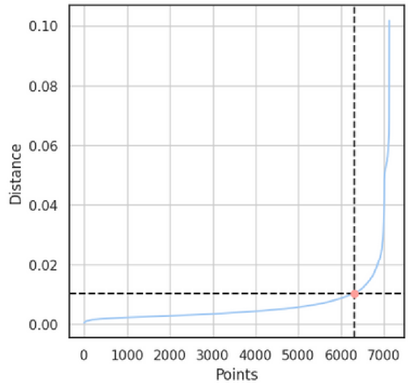
- **DBSCAN - Density-Based Spatial Clustering of Applications with Noise**
- **LOF - Local Outlier Fac-tor**
- **Auto - Encoder**
- **MCD - Minimum Covariance Determinant**

For most of these algorithms, we need to specify **the fraction of outliers (contamination)**.Therefore, as the first method, we choose **DBSCAN,** which does not require this parameter.

Submitted by Daniele Cecca

# DBSCAN

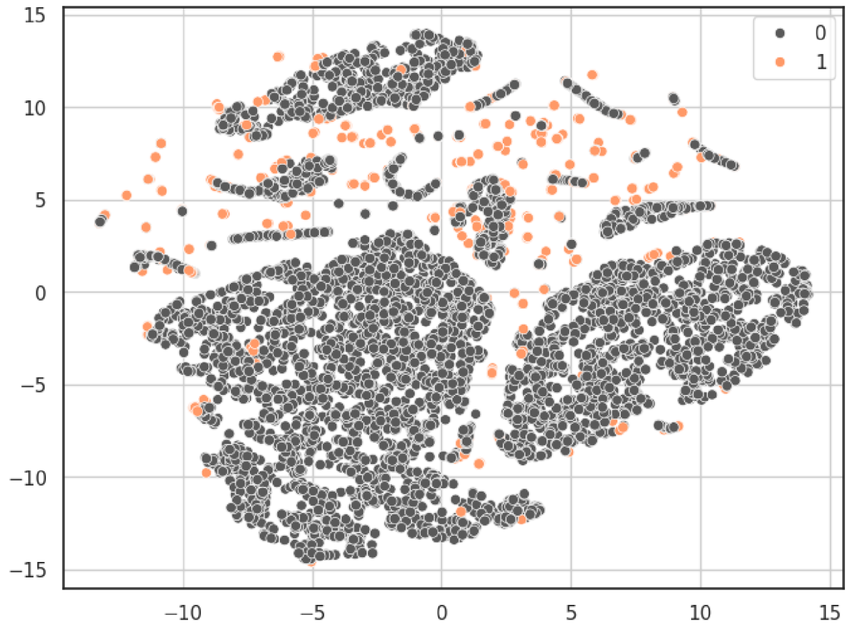
## DENSITY-BASED CLUSTERING




- **MinPts: 4**
- **Eps: 0.01**

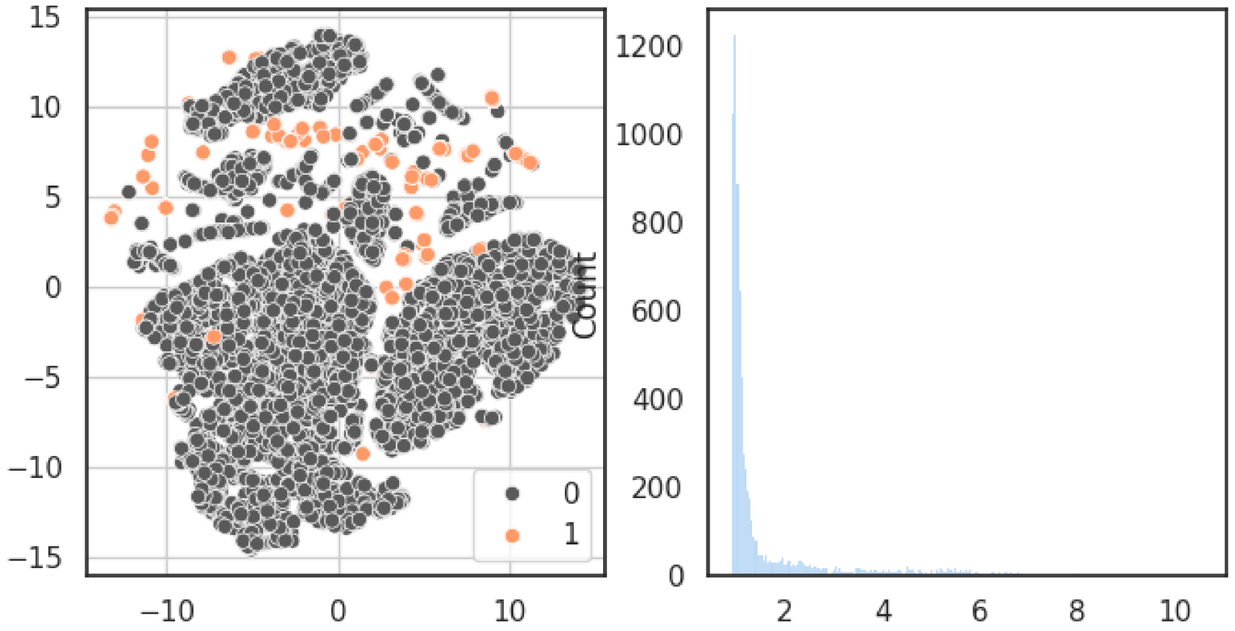
To determine **Eps**, we compute the distance from each point to its **4th nearest neighbo**r using **k-nearest neighbors (KNN),** and then sort these distances. As the distance metric, we will use the Gower distance. Therefore, we will reuse the proximity matrix computed previously.

Because **DBSCAN** found **640 outliers** we will use a **contamination of 10%** for the next models.



# LOF

## LOCAL OUTLIER FACTORS

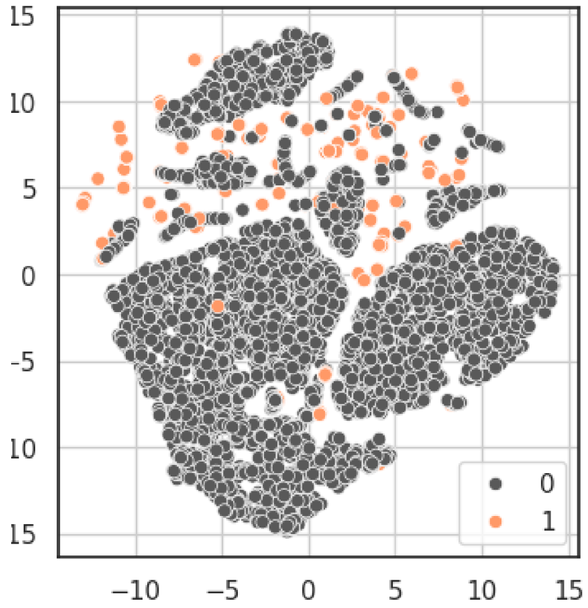


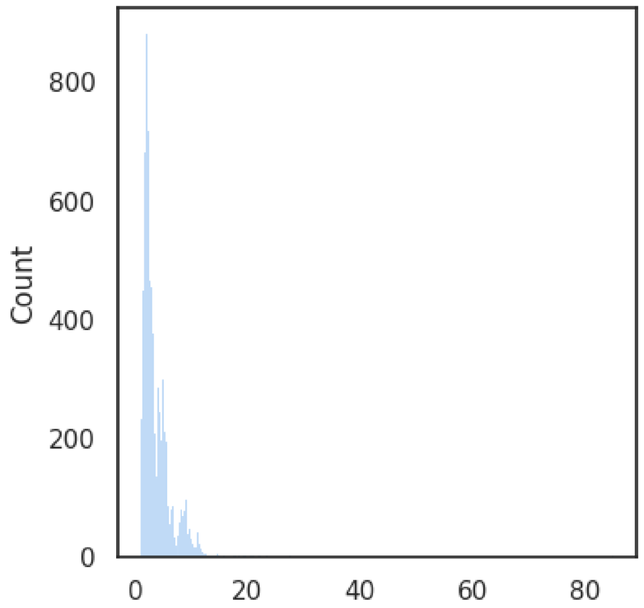

- **Number of normal data: 6422**
- **Number of outlier data: 707**

To choose the **number of neighbors**, we use the thumb rule, which suggests **taking 2% of the data**. As the distance metric, we will use again Gower.

# AE

## AUTO ENCODER



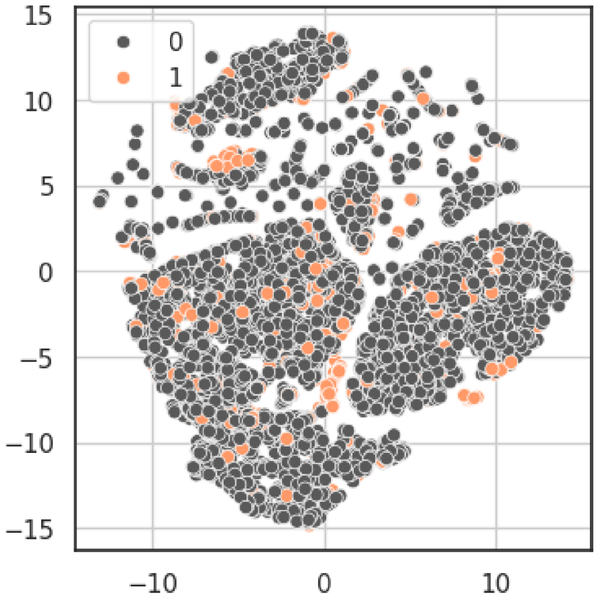



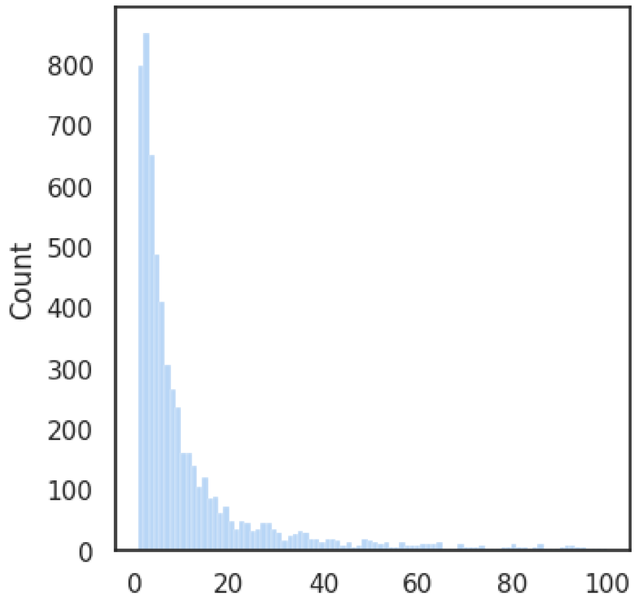

- **Number of normal data: 6416**
- **Number of outlier data: 713**

Similar to PCA, AE could be used to detect outlying objects in the data by **calculating the reconstruction errors.**

# MCD

## MINIMUM COVARIANCE DETERMINAT




- **Number of normal data: 6416**
- **Number of outlier data: 713**

It is a **statistical method** and could provide a different perspective on outliers.

The points flagged as outliers are **completely different** from those identified by the other methods.

# Check coherence between outlier

To check the difference between the models we compare the histograms by using the **mutual information** and we compare the labels by using **Jaccard score.**

To compare the scores, since they represent two different metrics, namely the **local outlier factor and reconstruction error**, we standardize the values by using **Min-Max S scaler**

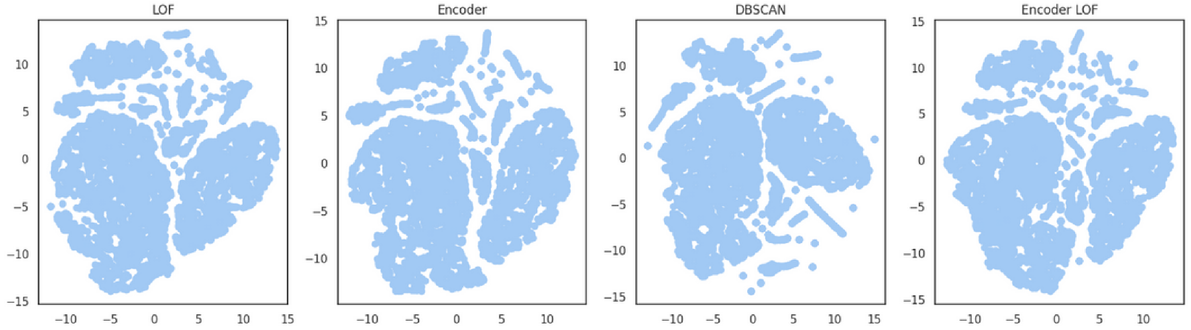
- **Mutual Information Score: 0.272**

- **Jaccard Coefficient LOF-Encoder : 0.56**
- **Jaccard Coefficient LOF-DBSCAN: 0.22**
- **Jaccard Coefficient Encoder-DBSCAN: 0.36**

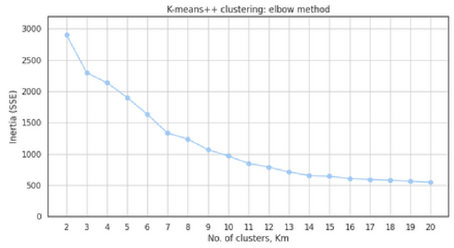
# Delete outliers

After the analysis of the different methods we **remove the outliers** from the dataset. and we create four different dataset:

1. **df_lof:** dataframe without the outliers found by LOF
2. **df_encoder:** dataframe without the outliers found by Auto-Encoder
3. **df_dbscan:** dataframe without the outliers found by DBSCAN
4. **df_encoder_lof:** dataframe without the outliers found in common between LOF and Auto-Encoder.



# Clustering





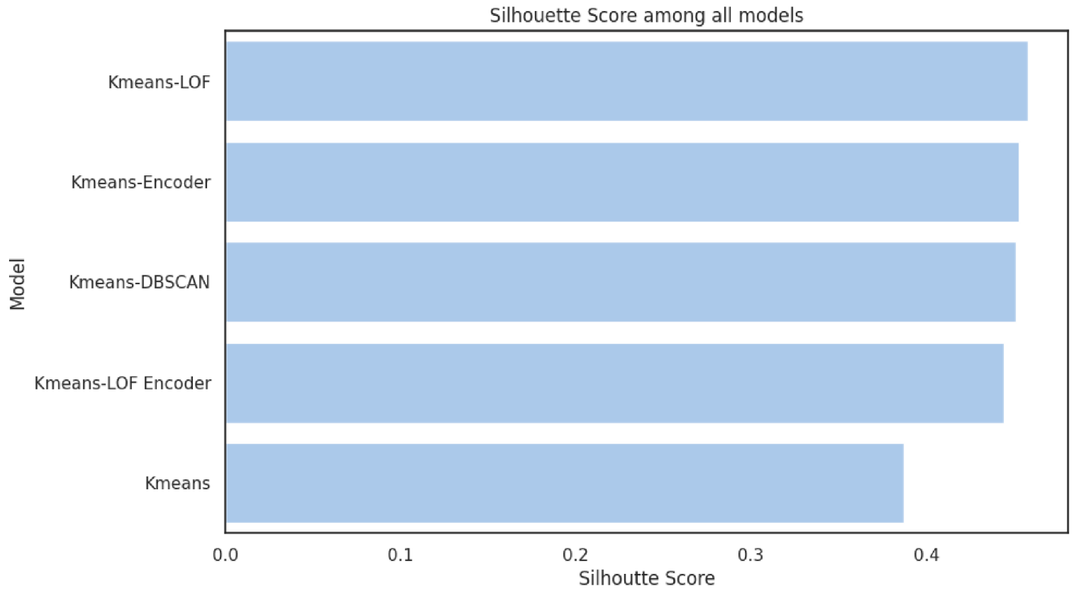
To evaluate the different methods of outlier detection, we apply k-means to the different dataframes and assess the results using the **Silhouette Score.**

Although our dataset may not be well-suited for k-means due to its small variance, we use this method only as a **reference point**. We **expect a poor result across all method**s, but we can still determine the best one

# COMPARISON

As anticipated, the results **are not very satisfactory.**
However, in all cases, we observe improved performance compared to the dataset including all data points.

The **LOF algorithm** demonstrates the best performance.




| Model | Silhouette Score |
|---|---|
| Kmeans-LOF | 0.458 |
| Kmeans-Encoder | 0.453 |
| Kmeans-DBSCAN | 0.451 |
| Kmeans-LOF Encoder | 0.444 |
| Kmeans | 0.387 |

# Probability

At the end, we decide to assign to each sample the probability of the data object being anomalous. We compute the following probability: **P(A = anomalous | X1, X2, . . . , Xn)**

To compute this score, we should use **Bayes' Theorem,** but we use **sigmoid function.**

We opted for the sigmoid function because it is **simpler** and does **not require knowledge of the score distribution.**

| Sample ID | Anomaly | Anomaly Probability |
|---|---|---|
| 3012 | 0 | 0.368795 |
| 6225 | 0 | 0.369883 |
| 5399 | 0 | 0.369970 |
| 4255 | 0 | 0.371542 |
| 405 | 1 | 0.903388 |
| 6999 | 0 | 0.515087 |
| 3419 | 0 | 0.379816 |
| 3715 | 0 | 0.374490 |
| 4803 | 0 | 0.677050 |
| 2472 | 0 | 0.379833 |

# FUTURE WORKS

Outlier detection often suffers from **model instability** due to its unsupervised nature, future work could involve **exploring other models** or trying **ensemble methods.**

Additionally, we could attempt to **build a Bayesian network** using the probabilities computed from the anomaly scores

# THANKS FOR ATTENTION

DANIELE CECCA MAT 918358